МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«НИЗКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ»

студента 2 курса, 23202 группы

Пятанова Михаила Юрьевича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Кандидат технических наук
Владислав Александрович
Перепёлкин

Новосибирск 2024

СОДЕРЖАНИЕ

ЦЕЛЬ

1. Ознакомиться с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb.

ОПИСАНИЕ РАБОТЫ

Была реализована программа, получающая список всех подключенных к машине USB устройств с использованием libusb (приложение 1). Для каждого найденного устройства были напечатаны его класс, идентификатор производителя и изделия, серийный номер, название производителя и устройства. Программа была запущена на виртуальной машине с установленной системой Ubuntu 24.04.1. Результат представлен в приложении 2.

Были изучены состав и характеристики обнаруженных USB устройств:

Название устройства	Класс	Идентификатор производителя	Идентификатор изделия	Серийный номер	Название производителя
EHCI Host Controller[1]	9 - Hub class[3]	0x1d6b	0x2	0000:00:0b.0	Linux 6.8.0-48-generic ehci_hcd
USB Tablet	0 - Per-Interface Class[4]	0x80ee	0x21	N/a	VirtualBox
OHCI PCI host controller[2]	9 - Hub class	0x1d6b	0x1	0000:00:06.0	Linux 6.8.0-48-generic ohci_hcd

1) The Enhanced Host Controller Interface (EHCI) – это USB контроллер, поддерживающий USB 2.0. Контроллеры EHCI предназначены для работы с высокоскоростными устройствами USB 2.0, которые работают со скоростью передачи данных до 480 Мбит/с.

2) The Open Host Controller Interface (OHCI) – это USB контроллер, поддерживающий устройства USB 1.1. Контроллеры OHCI предназначены для работы как с высокоскоростными (12 Мбит/с), так и с низкоскоростными (1,5 Мбит/с) USB-устройствами. Они обычно встречаются в старых системах и часто используются в сочетании с контроллерами EHCI для обеспечения всесторонней поддержки USB.

3) Устройство, которое расширяет один USB-порт на несколько портов, позволяя подключать несколько USB-устройств к одному хост-порту.

4) Некоторые устройства могут иметь несколько интерфейсов, каждый из которых служит для разных целей. Например, составное устройство может иметь один интерфейс для запоминающего устройства и другой для устройства связи.

ЗАКЛЮЧЕНИЕ

Получены начальные знания о библиотеке libusb, приобретен опыт низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах.

Приложение 1.

Команда для компиляции и запуска:
g++ src.cpp `pkg-config --libs --cflags libusb-1.0` -o e
sudo ./e

```
#include <iostream>
#include "libusb-1.0/libusb.h"

const int BUFFER_SIZE = 128;

void PrintDeviceInformation(libusb_device* device) {
    libusb_device_descriptor desc;
    int status = libusb_get_device_descriptor(device, &desc);
    if (status < 0) {
        std::cerr << "Error: failed to get device descriptor, code: " << status <<
std::endl;
        return;
    }
    // Output device class, vendor ID, and product ID
    std::cout << "Device class: " << (int) desc.bDeviceClass << std::endl;
    std::cout << "Vendor ID: " << std::hex << desc.idVendor << std::endl;
    std::cout << "Product ID: " << std::hex << desc.idProduct << std::endl;
    libusb_device_handle* handle;
    status = libusb_open(device, &handle);
    // Get and output the device's serial number
    if (status == 0) {
        // Buffers used to store the serial number
        unsigned char buffer[BUFFER_SIZE];
        // Get the device descriptor as a c-style string
        if (libusb_get_string_descriptor_ascii(handle, desc.iSerialNumber, buffer,
BUFFER_SIZE) > 0) {
            std::cout << "Serial number: " << buffer << std::endl;
        }
        if (libusb_get_string_descriptor_ascii(handle, desc.iManufacturer, buffer,
BUFFER_SIZE) > 0) {
            std::cout << "Manufacturer: " << buffer << std::endl;
        }
        if (libusb_get_string_descriptor_ascii(handle, desc.iProduct, buffer,
BUFFER_SIZE) > 0) {
            std::cout << "Product: " << buffer << std::endl;
        }
        libusb_close(handle);
    }
    else {
```

```
        std::cerr << "Exception code: " << status << std::endl;
        std::cerr << "Error: failed to open the device." << std::endl;
    }
    std::cout << std::endl;
}

int main() {
    // Libusb session context
    libusb_context* context = NULL;
    // Libusb init status
    int status;
    // Number of USB devices found
    ssize_t device_count;
    // Initialize libusb
    status = libusb_init(&context);
    if (status < 0) {
        std::cerr << "Error: failed to initialize libusb, code: " << status <<
std::endl;
        return 1;
    }
    // Get the list of connected USB devices
    // Pointer to pointer to device, used to obtain the list of devices
    libusb_device** devices;
    device_count = libusb_get_device_list(context, &devices);
    if (device_count < 0) {
        std::cerr << "Error: failed to get the list of devices." << std::endl;
        libusb_exit(context);
        return 1;
    }
    std::cout << "*** USB devices found: " << device_count << " ***" << std::endl;
    for (ssize_t i = 0; i < device_count; i++) {
        PrintDeviceInformation(devices[i]);
    }
    // Free resources
    libusb_free_device_list(devices, 1);
    libusb_exit(context);
    return 0;
}
```

Приложение 2.